Python con files:

La clase file:

Primero que todo, debemos saber que todo, absolutamente todo es Python, son objetos, clases, métodos y funciones. Los file, no son la excepción y tampoco los objetos del mismísimo SO.

Mientas más rápido entendamos quienes son los objetos, quien es la clase, y quienes son los métodos que estamos usando, en vez de solo escribir pura letra como gallinas sin cabeza antes entenderemos el código.

Una vez que sabemos esto, sabemos que lo primero que se hace cuando abrimos un archivo en Python, estamos creando una instancia, un objeto de la clase file, que tiene sus propios métodos, que lejos de complicarnos la vida, solo lo hacen más fácil. Esto se hace cuando decimos open(filename) as file, o cuando igualamos file, a open(filename); en ambos casos estamos creando el objeto file de la case file, al cual le podemos aplicar los métodos de esta clase

CSV reseña:

Programar con Python es bastante fácil, pero hacerlo mediante scripts, pequeños scripts, no siempre es una opción viable cuando estamos trabajando con grandes cantidades de datos. En estos casos utilizaremos los objetos del tipo file, o sea, archivos.

Un archivo se puede abrir, o acceder a él mediante la función open(). Dentro de esta, va lo que llamamos el path, o el directorio, que puede variar según el sistema operativo que estemos utilizando.

También [podemos asignar este archivo a una variable de dos maneras, igualando como normalmente haríamos o usando “with open (filename) as file:” .

Este archivo que hemos abierto puede contener mucha data a la que podemos acceder, y se me ocurren muchas formas de las cuales podría utilizarlo, pero…los creadores de Python parecen tener mejores formas que las que había pensado. Un archivo, de texto está conformado por líneas a las cuales podemos acceder para obtener una información determinada, así que lo mas inteligente, si queremos poner mucha data en un archivo es en la primera línea poner lo especial que tenga ese archivo y en las demás, el contenido de la data, por ejemplo si vamos a hacer un diccionario usando el CSV, que es una manera muy fácil de usar los archivos en la primera línea pondríamos las claves, y las demás serian valores de diferentes diccionarios anidados en una lista, por eso es importante aprender el CSV, porque es una manera de trabajar con la data demasiado fácil y sin tener que escribir tanto código.

Como ya había mencionado antes el enfoque de este documento será la importancia de aprender el modulo csv, y los tipos de archivo csv con los cuales podemos aplicar este módulo.

El archivo lo podemos abrir y modificarlo como vimos antes, con el módulo del OS, y ahora con el módulo de csv, vamos a poder acceder a información de este y también poderlo modificar o crear archivos con una estructura de información super fácil de utilizar y muy pero muy practica que es la lista de diccionarios mediante el módulo csv.

Métodos de la clase file :

Primero la función que usamos para abrir el archivo no pertenece al módulo csv, sino a Python, así que no necesitamos hacer uso de import csv para poder tener acceso a dicha función, así como los demás métodos que vamos a utilizar en este párrafo.

file.readline(): Este método lee una línea a la vez como se fuera un for, es muy práctico para este uso, ya que podemos recorrer cade una de las líneas como una iteración y además podemos salvarlo en una variable, como una lista, por ejemplo.  
file.read(): Lee todas las líneas que le siguen a la línea actual, si es la primera vez que se usa o si no se ha usado el file.readline(), leerá todo el archivo file.close(): Se usa para cerrar el file, SIEMPRE hay que cerrarlo.

Recordar que cada línea, es separada e interpretada dentro de Python por /n, así que cada una de las líneas al final siempre tendrá este carácter, así que si le realizas un for, y aplicas el método upper para la clase string por ejemplo, te devolverá una línea vacía entre cada línea, para eliminar esto, solo debemos usar el strip que se hará cargo de todos los espacios ya sean nuevas líneas o tabulaciones.

file.write(): Se usa para poder escribir dentro de un archivo, para que este funcione debemos tener de antemano permiso de escritura “w”,”a”,”r+”.

El módulo OS:

Como habíamos dicho, para trabajar con archivos dentro del sistema también podemos usar los métodos que provee el módulo OS, o sea el propio Sistema Operativo, que no es más que un módulo llena de métodos para hacer el trabajo más fácil y como módulo al fin, antes de poder utilizarlo, tenemos que importarlo para poder utilizarlo. La peculiaridad de estos métodos es que debemos tener cuidado con los directorios o caminos, normalmente cuando usamos diferentes OS los directorios cambian. Así que, si vas a eliminar o mover un archivo, tienes que obviamente estar de antemano en el directorio donde se encuentra el archivo, o debes de cambiar de directorio, con uno de los métodos del OS o debes paralo el directorio absoluto de donde es que se encuentra el archivo a eliminar o de lo contrario podrías perder archivos valiosos.

Métodos de la clase OS:

os.remove(“C:\filename”):Remueve un archivo del sistema, eliminándolo por completo, a este método se le pasa el nombre del archivo, en caso de estar dentro del directorio en el que estamos, o en caso de no estarlo, le pasamos directamente el directorio absoluto del archivo que deseamos eliminar.

os.rename(“old name”, “new name”): Sirve para cambiar el nombre del archivo.

os.path: Este modulo contiene una amplia variedad de submódulos con que se pueden hacer muchas cosas.

os.path.exists(“filename”): Devuelve un booleano de si el archivo existe o no en ese directorio.

os.path.getsize(“filename”): Devuelve el tamaño del archivo en bytes.

os.path.join(directorio, archivo o directorio): A primera vista parece una función estúpida ya que esto lo podemos hacer manualmente introduciendo un /, pero el problema es que la manera de separar directorios dentro de los sistemas operativos varia, en Linus y Mac se hace con “/”, mientras que en Windows se hace con”\”; la función se ocupa de que no importa en que OS corra nuestro script, el código corra sin ningún tipo de error.

os.path.getmtime(“filename”) Devuelve la cantidad de segundos desde una fecha timestamp hasta la fecha que fue modificada, podemos convertirlo en una fecha normal haciendo uso del modulo datetime, dentro de este, usar el metodo datetime, y luego dentro de este, el metodo timestamp así:

variable\_temporal= os.path.getmtime(“filename”) datetime.datetime.fromtimestamp(variable\_temporal)

os.getcwd:Nos devuelve el directorio absoluto de donde estamos

os.mkdir(“C:\Users\Temporal”): hace un nuevo directorio en el directorio en el que estamos

os.chdir(“C:\Users\Mike Ehl”): cambia a otro directorio

os.rmdir(“C:\Users\Temporal”): Elimina un directorio, si el directorio esta vacio, si el directorio contiene algún archivo, el directorio no podrá ser eliminado y obtendremos un error.

os.listdir(“C:\Users”): Crea una lista de los directorios y archivos dentro del directorio seleccionado.

os.

Trabajando con CSV:

Primeramente, para trabajar con el script del comando del sistema operativo, hay que escribir perfectamente como es que se va a ejecutar el código, de no ser así habrá que escribir todo el código desde cero.

Los CSV files, son tipos de archivos, los “tipos de archivos” no son mas que archivos, que obviamente contienen textos pero que existe un módulo, o un programa entero dedicado a leer este tipo de archivo específicamente por ejemplo los HTML o los AVI o los JPG. Cada uno de estos archivos se usa para plasmar un tipo de información diferente, los CSV o “Coma Separated Values” son otro tipo de archivo.

Estos tipos de archivos nos permiten acceder a información de manera fácil ahorrándonos mucho código con solo acceder al modulo. Estos archivos se leen y se escriben por líneas de archivos, donde la primera línea, es un valor key, de un diccionario, y las demás líneas corresponden a los valores correspondientes a esa clave, para cada uno de los diccionarios que corresponden a cada una de las líneas del archivo respectivamente. O sea, el modulo CSV, crea un archivo que se puede leer como una lista donde los elementos de la lista, son diccionarios. Esto es muy práctico cuando queremos salvar diferentes tipos de datos, que van a ser los mismos, paras diferentes instancias, por ejemplo, una instancia sería una persona, y las palabras claves, o los tipos de datos que queremos salvar en el diccionario seria datos sobre esa persona, por ejemplo, su nombre, apellido, numero de teléfono, rol que ocupa en la compañía etc. y los valores, que serian los datos correspondientes a cada una de las claves.

El módulo CSV viene con diferentes métodos que podemos utilizar para crear, modificar, agregar máa elementos, eliminar, y leer estos diccionarios en la lista.

Creando un CSV:

Para crear un archivo del tipo csv, el cual lee su contenido como una lista anidada de listas, o una lista anidada de diccionarios, debemos por supuesto de antemano guardar en una variable una lista anidada de listas o una lista anidada de diccionario con los datos que deseamos guardar en el archivo para luego pasarle uno de los métodos de escritura del modulo CSV, y este hará el resto del trabajo. Hay dos métodos principales que se utilizan para escribir en el CSV, uno se utiliza para escribir todos los rows al mismo tiempo, y el otro para escribir un solo row, o sea para agregar uno nuevo.

Antes de pasarle el metodo csv.writerow o el metodo csv.writerows debemos declarar una variable, la cual será igualada al pasarle el metodo csv.write a la variable a la que hicimos igual a abrir el archivo, o crearlo, si este no existía. Esto se hace porque los métodos csv.writerow y csv.writerows, solo se pueden llamar a una instancia de la clase write, dentro del modulo csv, y no dentro de una variable normal, porque recordemos que solamente hacer igual una variable a abrir el archivo, no es una instancia del módulo csv, y si queremos usar los métodos de escritura por row, es lógico pensar que necesitaremos una instancia de una clase dentro de este modulo.

Hay que recordar que una vez creemos el archivo la extensión de este debe ser csv y no txt.

Texto

Descripción generada automáticamente

Métodos del módulo CSV:

csv.reader(file):Crea una instancia de la clase reader del módulo csv, a través de la cual se pude iterar como una lista.

csv.DictReader(file):Crea una instancia de la clase DictReader del módulo csv, a través de la cual se pude iterar como un diccionario, este diccionario se crea con los elementos del primer row de la lista como keys y los elementos de las demás listas como valores para cada uno de los diccionarios que son generados para cada una de las líneas. Este tiene la ventaja que para acceder a uno de sus valores, no necesitamos saber de antemano la posición del index del valor que estamos buscando, porque como es un diccionario solamente tenemos que pasarle a la instancia de la clase DictReader que hemos creado, la key que necesitamos.

csv.writerow(file): Cuando es aplicado a una instancia de la clase write, a la que se le ha pasado una lista escribe un row en el archivo. El archivo debe estar abierto con permisos de escritura.

csv.writerows(file): Cuando es aplicado a una instancia de la clase write, a la que se le ha pasado una lista anidada de listas, escribe los rows en el archivo. El archivo debe estar abierto con permisos de escritura.

Para Escribir como diccionarios:

Texto

Descripción generada automáticamente

csv.DictWriter: Nos da una instancia de la clase DictWriter del modulo csv, con el cual podemos crear una lista anidada de diccionarios, ejemplo: writer. A esta instancia se le deben de pasar dos parámetros, el primero, es como siempre la variable en la que hemos guardado el archivo, y la segunda son las keys, que deben de haber sido guardadas de antemano en una lista.

writer.writeheader(): sirve para definir las keys de los diccionarios que serán almacenados en la lista de diccionarios.

writer.writerows(Dict): sirve para definir los diccionarios que vamos a agregar la lista de diccionarios Dict, que es una variable que almacena una lista de diccionarios.